CLAIMS

1-5. (Canceled)

6. (Original) A data management system, said system characterized as a composite system, the system comprising a plurality of processes;

each process having an interface and implementing at least one respective service defined by that interface;

a first invocation of the at least one respective service by a transaction resulting in the creation of a first transaction local to the process thereof, the first local transaction being a child of the invoking transaction and being parent of any transaction triggered by invocation of a service of another process;

a second invocation of the at least one respective service by a transaction resulting in the creation of a second transaction local to the process thereof, the first local transaction being a child of the invoking transaction and being parent of any transaction triggered by invocation of a service of another process;

each process characterized in that if the first transaction and the second transaction conflict but are both children of a same invoking transaction, then the first transaction and the second transaction are not executed concurrently;

each process further characterized in that each transaction local thereto is independently handled at the process;

each process making scheduling and recovery decisions independent of any centralized component.

7. (Original) The system of claim 6 wherein the root transaction is able to dynamically set concurrency preferences for the resulting distributed transaction, based on client needs.

8. (Amended) A data management system, said system characterized as a composite system, the system comprising a plurality of processes;

each process having an interface and implementing at least one respective service defined by that interface;

invocation of the at least one respective service by a [trand] thread of the invoking transaction and being parent of any transaction triggered by invocation of a service of another process;

each process further characterized in that each transaction local thereto is independently handled at the process;

each process making scheduling and recovery decisions independent of any centralized component. triggered by invocation of a service of another process, each process further characterized in that each transaction local thereto is independently handled at the process, each process making scheduling and recovery decisions independent of any centralized component, the method comprising the steps of:

propagating from a first process to a second process a message indicative of a globalCommit operation with respect to a root transaction, said message also indicative of a number or identifying list of invocations which the first process has made to the second process on behalf of the root transaction;

within the second process, comparing the number or list indicated in the message with a count or list within the second process of the number or list of invocations which have been made on behalf of the root transaction;

in the event the comparison yields a non-match, aborting the transaction.

9. (Original) The system of claim 8 wherein each process is built using Java.

10. (Original) A method for use with a data management system, said system characterized as a composite system, the system comprising a plurality of processes, each process having an interface and implementing at least one respective service defined by that interface, invocation of the at least one respective service by a transaction resulting in the creation of a transaction local to the process thereof, the local transaction being a child of the invoking transaction and being parent of any transaction triggered by invocation of a service of another process, each process further characterized in that each transaction local thereto is independently handled at the process, each process making scheduling and recovery decisions independent of any centralized component, the method comprising the steps of:

propagating from a first process to a second process a message indicative of a globalCommit operation with respect to a root transaction, said message also indicative of a number or list of invocations which the first process has made to the second process on behalf of the root transaction;

within the second process, comparing the number or list indicated in the message with a count or list within the second process of the number or list of invocations which have been made on behalf of the root transaction;

in the event the comparison yields a match, proceeding with the globalCommit operation.

11. (Original) A method for use with a data management system, said system characterized as a composite system, the system comprising a plurality of processes, each process having an interface and implementing at least one respective service defined by that interface, invocation of the at least one respective service by a transaction resulting in the creation of a transaction local

to the process thereof, the local transaction being a child of the invoking transaction and being parent of any transaction triggered by invocation of a service of another process, each process further characterized in that each transaction local thereto is independently handled at the process, each process making scheduling and recovery decisions independent of any centralized component, the method comprising the steps of:

propagating from a first process to a second process a message indicative of a globalCommit operation with respect to a root transaction, said message also indicative of a number or list of invocations which the first process has made to the second process on behalf of the root transaction;

within the second process, comparing the number or list indicated in the message with a count or list within the second process of the number or list of invocations which have been made on behalf of the root transaction;

in the event the comparison yields a non-match, aborting the transaction.

12. (Amended) A distributed system, said system characterized as a composite system, the system comprising a plurality of processes;

each process having an interface and implementing at least one respective service defined by that interface;

each or any globalCommit message exchange between processes also carrying information about the actual work being committed [(] or being agreed upon[)].

13. (Amended) The system of claim 12, such information being logged for recoverability in the event of a crash, such information being used for assistance [(] or presented [)] at any time before, during or after global commitment.

14. (Original) The system of claim 12 or 13, wherein any globalCommit requires a registration, and wherein the registration for a globalCommit also carries information about the actual work being committed.

15. (Amended) A method for use in a distributed system, said system characterized as a composite system, the system comprising a plurality of processes, each process having an interface and implementing at least one respective service defined by that interface, the method comprising the step of:

for each globalCommit message exchanged between processes, including also information about the actual work being committed [(] or being agreed upon [)].

16. (Amended) The method of claim 15 further comprising the step of logging such information for recoverability in the event of a crash, such information being used for assistance [(] or presented [)] at any time before, during or after global commitment.

17. (Original) The method of claim 15 or 16 further comprising the step of propagating a registration for a globalCommit, wherein the registration for a globalCommit also carries information about the actual work being committed.

18. (Amended) A distributed system, said system characterized as a composite system, the system comprising a plurality of processes;

each process having an interface and implementing at least one respective service defined by that interface;

wherein the root invocation or [(] transaction [)] or, alternatively, the root's human user is allowed to dynamically set its/his concurrency preferences for the entire invocation.

19. (Amended) The system of claim 18 wherein the root invocation (transaction) propagates the concurrency preferences with each or any child invocation it makes.

20. (Original) The system of claim 19 wherein each invocation propagates the concurrency preferences as it has received them from the root invocation.

21. (Amended) The system of claim 20 wherein the propagated concurrency preferences [(] at any level in the root invocation's invocation hierarchy [)] specify the extent to which shared resource access is desired or allowed or denied among descendant invocations of the root invocation [(] or user [)] and other, concurrent invocations who are also descendants of the same root.

22. (Amended) The system of claim 20 wherein the propagated concurrency preferences [(] at any level in the root invocation's invocation hierarchy [)] specify the extent to which shared resource access is desired or allowed or denied among descendant invocations of the root invocation [(] or user [)] and other, concurrent invocations who are not descendants of the same root.

23. (Amended) A data management system, referred to as service, comprising:

·One or more operations that can be invoked by remote clients;

·Some or all such remote clients having one or more associated contexts or transaction contexts;

·An invocation by a remote client also containing partial or complete information indicating or containing said client"s context or contexts;

·An invocation, by a remote client, of an operation leading to a new transaction different from, but possibly related to, any existing client transaction;

·Such an operation-level transaction being committed before the client context is terminated [(] before globalCommit notification [)];

·The service maintaining an undo operation for such a committed operation;

·A failing or failed remote client context leading to the execution of the undo operations of the corresponding committed invocations in the service.

24. (Original) The system of claim 23 where some or all undo operations are executed in an order that is the reverse of the order of their original counterparts.

25. (Original) The system of claim 23 where in addition the undo operations are chosen or defined in the same system as the one where the corresponding normal operations were executed.

26. (Original) The system of claim 23 where some or all undo operations are unknown to a remote client or its context.

27. (Original) The system of claim 23 where some or all undo operations are executed after a timeout and independent of whether the client's context outcome requires such undo.

28. (Original) The system of claim 23 where an undo operation's effects are confined to the data managed by the service on which the undo operation is maintained, even if the original operation involved other services.

29. (Original) The system of claim 23 where the service keeps locks to ensure that undo operations can be executed correctly.

30. (Original) The system of claim 23 where client context-related information is also part of any global commit message exchanges.

31. (Original) The system of claim 23 where client context information includes application-specific data.

32. (Amended) The system of claim 31 where [client] all or part of the context information is logged, i.e. stored on persistent storage, and retrievable by a human administrator.

33. (Original) The system of claim 23 where the service accepts messages indicative of which previously committed operations have to be undone.

34. (Original) The system of claim 23 where the service accepts messages indicative of which previously committed operations do not have to be undone.

35. (Original) The system of claim1 where some or all invocations are message-based or asynchronous.

36. (Original) The system of claim 23 where some or all invocations are synchronous.

37. (Original) The system of claim 23 where the client can request the undo executions of its invocations at the service while still allowing globalCommit in the end.